EXHIBIT NO. 4

Dockets.Justia.com

US006768999B2

(12) **United States Patent**
Prager et al.

(10) **Patent No.: US 6,768,999 B2**
(45) **Date of Patent: Jul. 27, 2004**

(54) **ENTERPRISE, STREAM-BASED, INFORMATION MANAGEMENT SYSTEM**

(75) Inventors: **Randy Prager**, New York, NY (US); **Peter Sparago**, Cheshire, CT (US)

(73) Assignee: **Mirror Worlds Technologies, Inc.**, New Haven, CT (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 457 days.

(21) Appl. No.: **09/892,258**

(22) Filed: **Jun. 26, 2001**

(65) **Prior Publication Data**

US 2002/0055946 A1 May 9, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/398,611, filed on Sep. 17, 1999, now Pat. No. 6,638,313, which is a continuation of application No. 08/673,255, filed on Jun. 28, 1996, now Pat. No. 6,006,227.

(60) Provisional application No. 60/274,575, filed on Mar. 9, 2001, and provisional application No. 60/240,480, filed on Oct. 13, 2000.

(51) **Int. Cl.**[7] **G06F 17/30**

(52) **U.S. Cl.** **707/102**; 707/2; 715/500.1; 715/526; 345/836

(58) **Field of Search** 707/102, 2; 715/500.1, 715/526; 345/838

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,867,494	A		2/1999	Krishnaswamy et al.	
5,867,665	A		2/1999	Butman et al.	
5,913,024	A		6/1999	Green et al.	
5,918,018	A		6/1999	Gooderun et al.	
6,006,227	A		12/1999	Freeman et al.	
6,240,429	B1	*	5/2001	Thornton et al.	715/500
6,243,724	B1		6/2001	Mander et al.	
6,466,237	B1	*	10/2002	Miyao et al.	345/838
6,523,048	B2	*	2/2003	DeStefano	715/526
6,638,313	B1	*	10/2003	Freeman et al.	715/500.1

OTHER PUBLICATIONS

User Guide Published by Mirror Worlds Technologies, Inc. of New Haven, Connecticut.

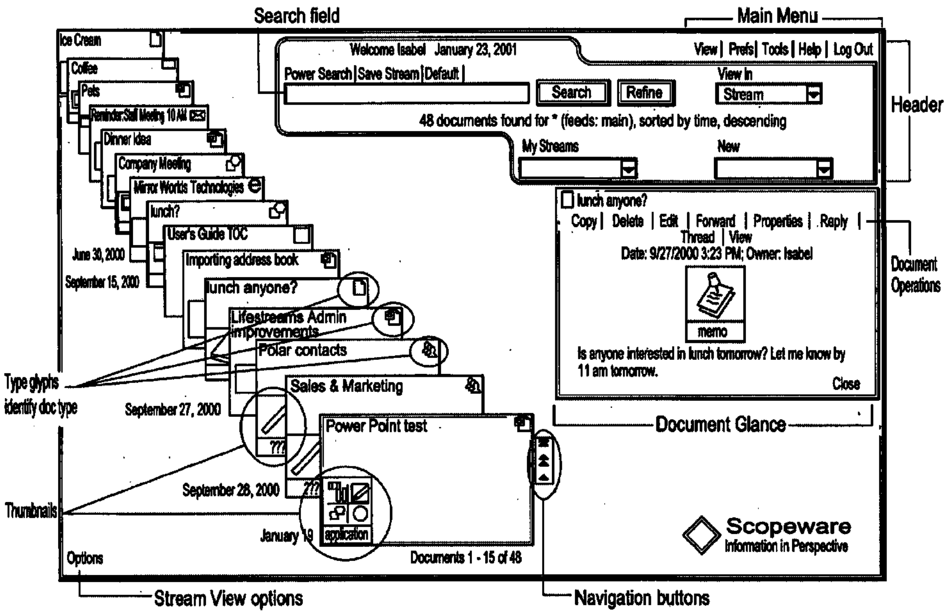
* cited by examiner

Primary Examiner—Wayne Amsbury
(74) *Attorney, Agent, or Firm*—Cooper & Dunham LLP

(57) **ABSTRACT**

Disclosed is a computer program product and method that operate an enterprise information system of at least one server and a number or personal computers communicating with each other and with the server. The program product and method create object models that have a consistent structure from and about information assets that are of diverse types and come from diverse software, and display browse cards about the information assets in a time-ordered stream, together with glance views related to the document object models. The glance views are displayed essentially in real time in response to passing a cursor over respective browse cards on the display.

1 Claim, 9 Drawing Sheets


Search field
Main Menu
Header
Welcome Isabel January 23, 2001
View | Prefs | Tools | Help | Log Out
Power Search | Save Stream | Default
Search
Refine
View In
Stream
48 documents found for * (feeds: main), sorted by time, descending
My Streams
New
lunch anyone?
Copy | Delete | Edit | Forward | Properties | Reply | Thread | View
Date: 9/27/2000 3:23 PM; Owner: Isabel
memo
Is anyone interested in lunch tomorrow? Let me know by 11 am tomorrow.
Close
Document Operations
Document Glance
Type glyphs identify doc type
Thumbnails
Options
Documents 1 - 15 of 48
Scopeware
Information in Perspective
Stream View options
Navigation buttons

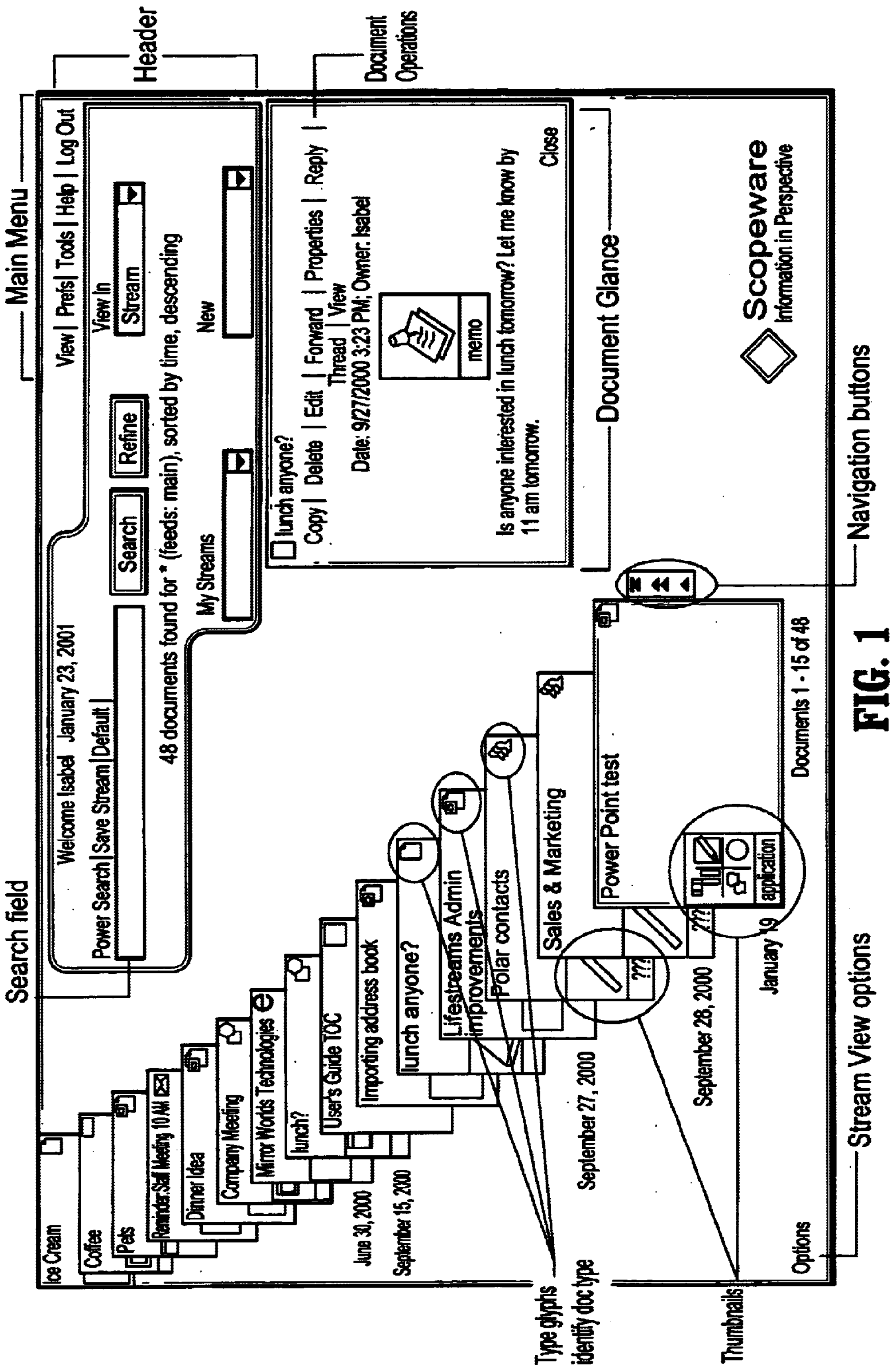

Main Menu
Header
Document Operations
Search field
View | Prefs | Tools | Help | Log Out
Welcome Isabel January 23, 2001
Power Search | Save Stream | Default
Search
Refine
View In
Stream
48 documents found for * (feeds: main), sorted by time, descending
My Streams
New
lunch anyone?
Copy | Delete | Edit | Forward | Properties | Reply
Thread | View
Date: 9/27/2000 3:23 PM; Owner: Isabel
memo
Is anyone interested in lunch tomorrow? Let me know by 11 am tomorrow.
Close
Document Glance
Scopeware
Information in Perspective
Navigation buttons
Ice Cream
Coffee
Pets
Reminder: Staff Meeting 10 AM
Dinner Idea
Company Meeting
Mirror Worlds Technologies
lunch?
User's Guide TOC
Importing address book
lunch anyone?
Lifestreams Admin improvements
Polar contacts
Sales & Marketing
Power Point test
application
???
June 30, 2000
September 15, 2000
September 27, 2000
September 28, 2000
January 19
Documents 1 - 15 of 48
Options
Stream View options
Type glyphs identify doc type
Thumbnails

FIG. 1

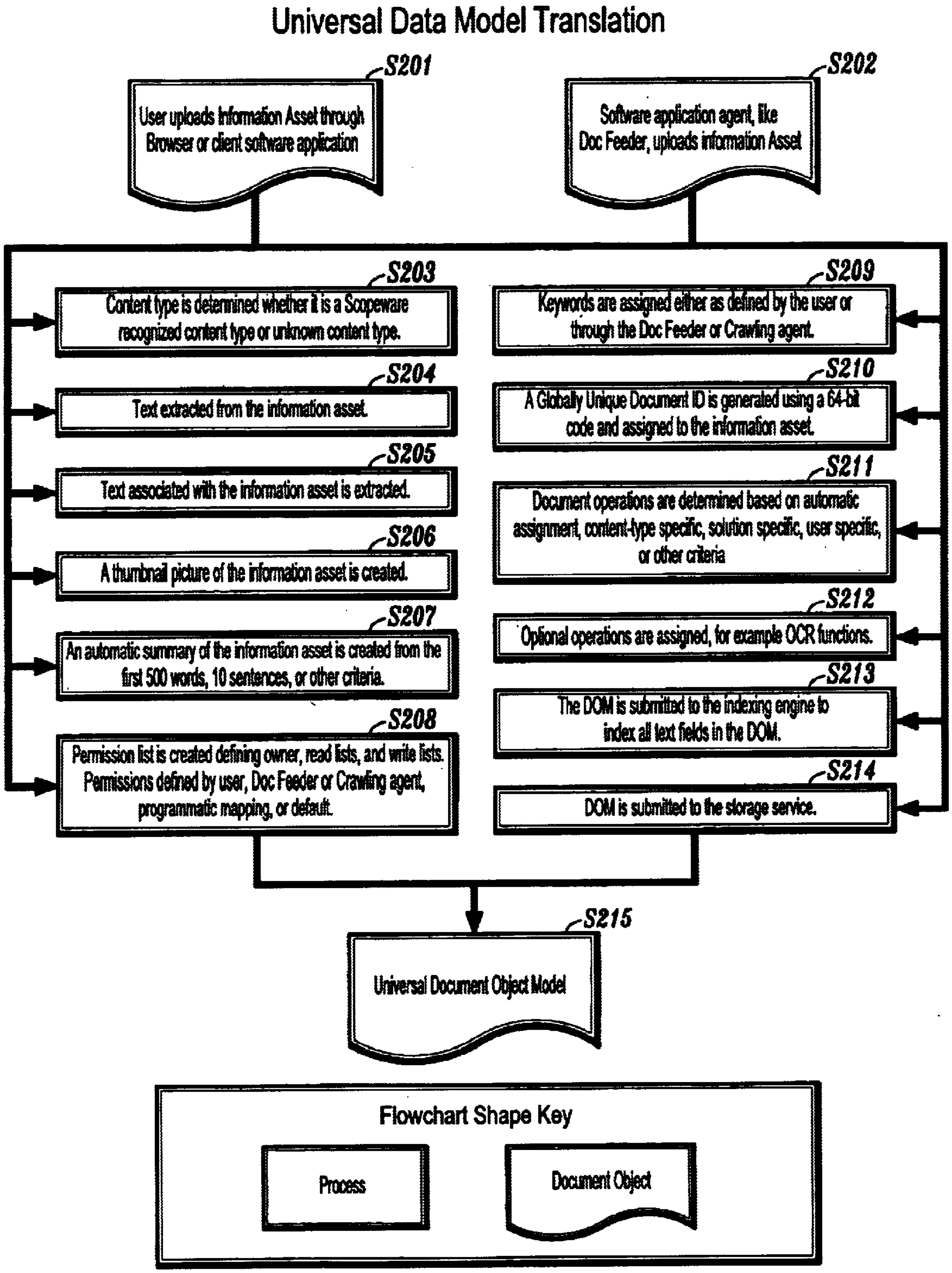

Universal Data Model Translation
S201
User uploads Information Asset through Browser or client software application
S202
Software application agent, like Doc Feeder, uploads information Asset
S203
Content type is determined whether it is a Scopeware recognized content type or unknown content type.
S204
Text extracted from the information asset.
S205
Text associated with the information asset is extracted.
S206
A thumbnail picture of the information asset is created.
S207
An automatic summary of the information asset is created from the first 500 words, 10 sentences, or other criteria.
S208
Permission list is created defining owner, read lists, and write lists. Permissions defined by user, Doc Feeder or Crawling agent, programmatic mapping, or default.
S209
Keywords are assigned either as defined by the user or through the Doc Feeder or Crawling agent.
S210
A Globally Unique Document ID is generated using a 64-bit code and assigned to the information asset.
S211
Document operations are determined based on automatic assignment, content-type specific, solution specific, user specific, or other criteria
S212
Optional operations are assigned, for example OCR functions.
S213
The DOM is submitted to the indexing engine to index all text fields in the DOM.
S214
DOM is submitted to the storage service.
S215
Universal Document Object Model
Flowchart Shape Key
Process
Document Object

FIG. 2

Scopeware Agents are Translators into Scopeware Document Object Models

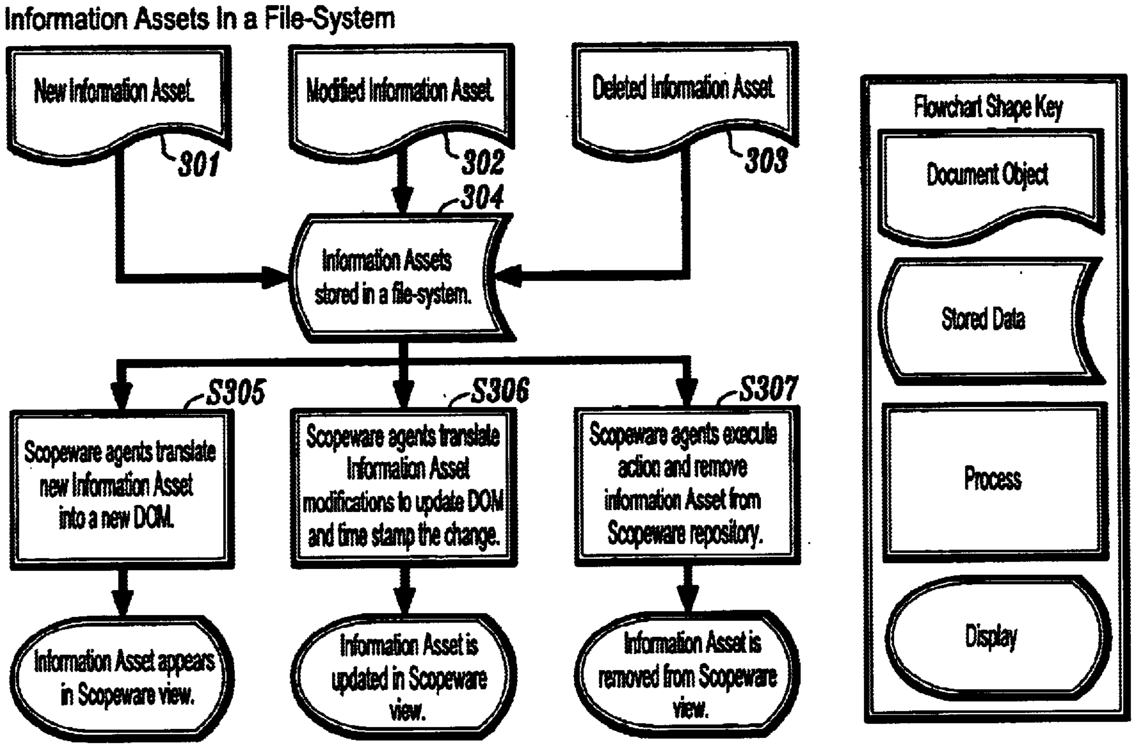

Information Assets In a File-System
New Information Asset
301
Modified Information Asset
302
Deleted Information Asset
303
304
Information Assets stored in a file-system.
S305
S306
S307
Scopeware agents translate new Information Asset into a new DOM.
Scopeware agents translate Information Asset modifications to update DOM and time stamp the change.
Scopeware agents execute action and remove information Asset from Scopeware repository.
Information Asset appears in Scopeware view.
Information Asset is updated in Scopeware view.
Information Asset is removed from Scopeware view.
Flowchart Shape Key
Document Object
Stored Data
Process
Display

FIG. 3

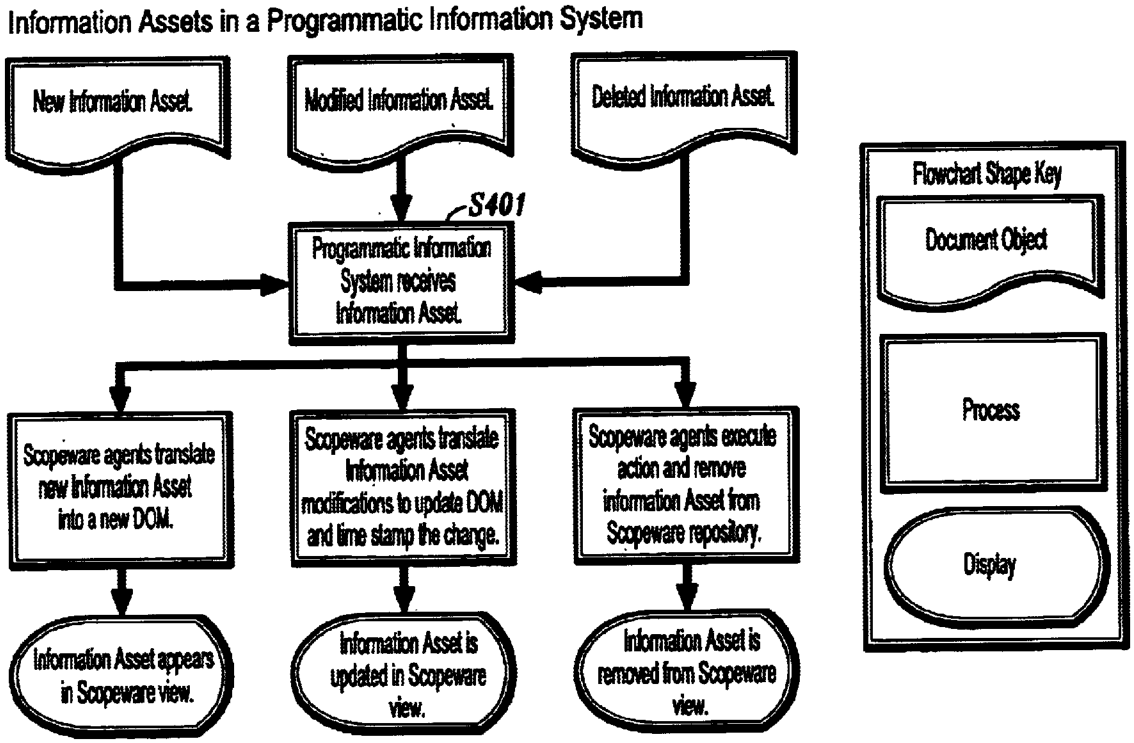

Information Assets in a Programmatic Information System
New Information Asset
Modified Information Asset
Deleted Information Asset
S401
Programmatic Information System receives Information Asset.
Scopeware agents translate new Information Asset into a new DOM.
Scopeware agents translate Information Asset modifications to update DOM and time stamp the change.
Scopeware agents execute action and remove information Asset from Scopeware repository.
Information Asset appears in Scopeware view.
Information Asset is updated in Scopeware view.
Information Asset is removed from Scopeware view.
Flowchart Shape Key
Document Object
Process
Display

FIG. 4

Instantaneously Dynamic, Tailored, and Interactive Document Glance Views

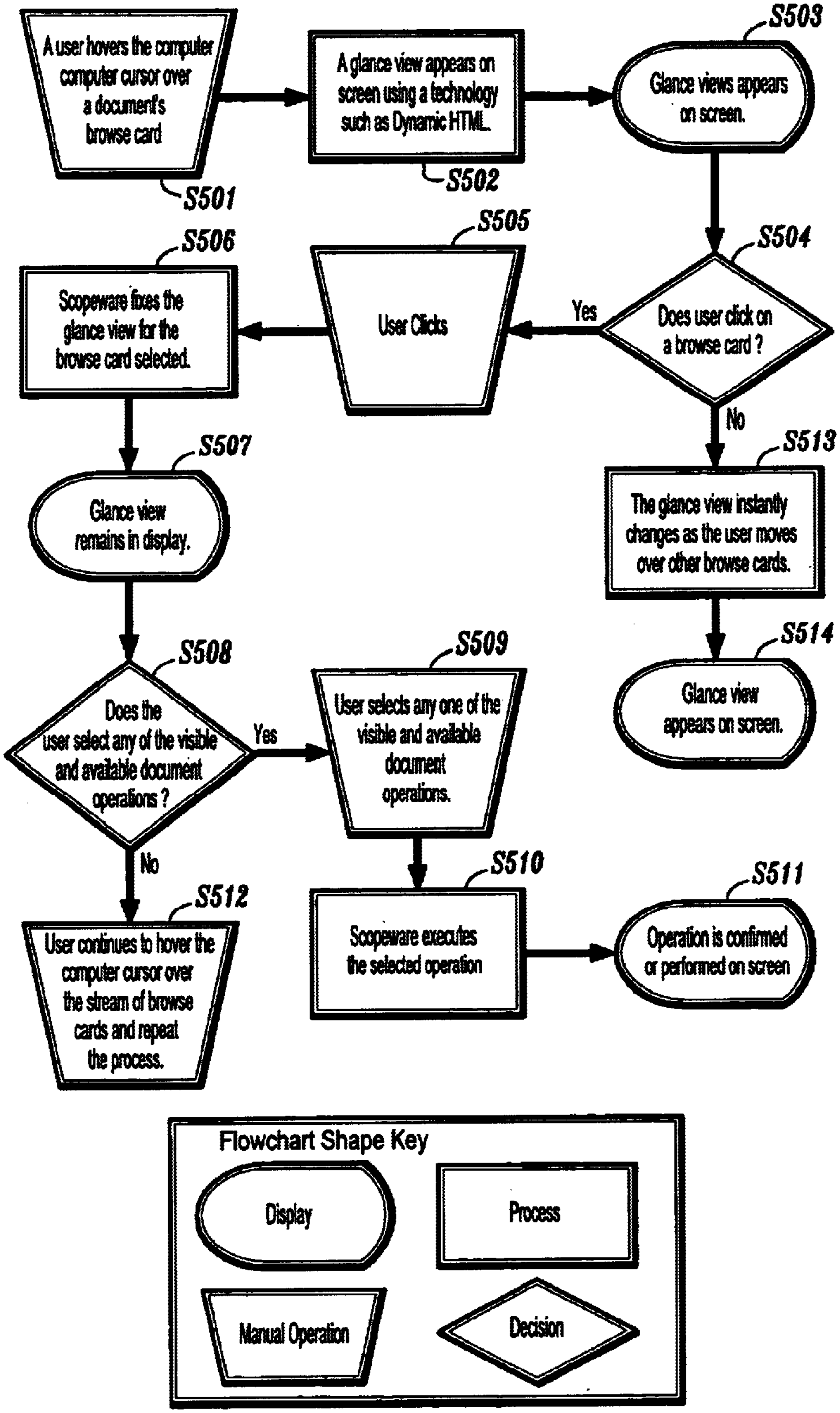

A user hovers the computer computer cursor over a document's browse card
S501
A glance view appears on screen using a technology such as Dynamic HTML.
S502
Glance views appears on screen.
S503
Does user click on a browse card ?
S504
Yes
No
User Clicks
S505
Scopeware fixes the glance view for the browse card selected.
S506
Glance view remains in display.
S507
Does the user select any of the visible and available document operations ?
S508
Yes
No
User selects any one of the visible and available document operations.
S509
Scopeware executes the selected operation
S510
Operation is confirmed or performed on screen
S511
User continues to hover the computer cursor over the stream of browse cards and repeat the process.
S512
The glance view instantly changes as the user moves over other browse cards.
S513
Glance view appears on screen.
S514
Flowchart Shape Key
Display
Process
Manual Operation
Decision

FIG. 5

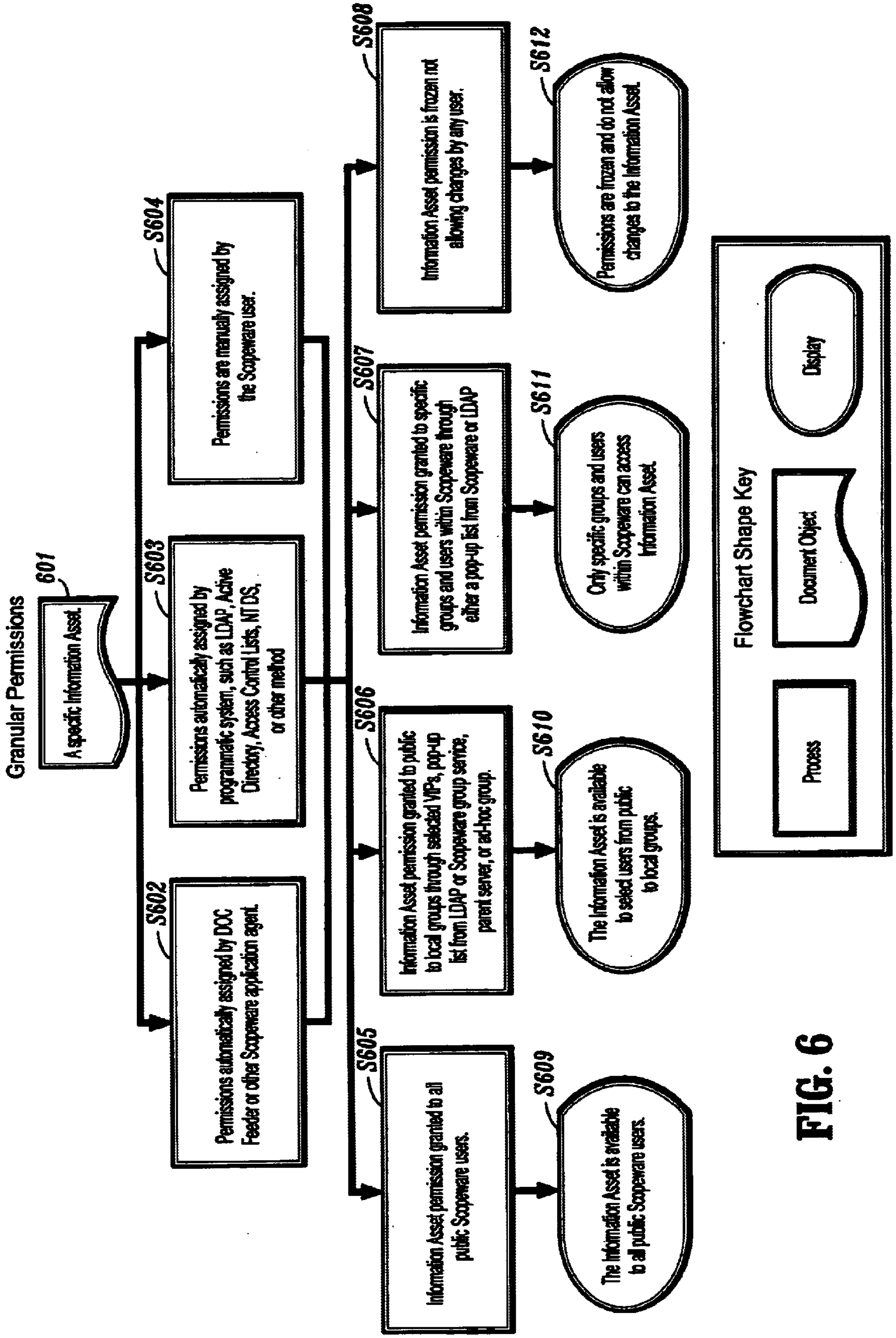

Granular Permissions
601
A specific Information Asset.
S602
Permissions automatically assigned by DOC Feeder or other Scopeware application agent.
S603
Permissions automatically assigned by programmatic system, such as LDAP, Active Directory, Access Control Lists, NT DS, or other method
S604
Permissions are manually assigned by the Scopeware user.
S605
Information Asset permission granted to all public Scopeware users.
S606
Information Asset permission granted to public to local groups through selected VIPs, pop-up list from LDAP or Scopeware group service, parent server, or ad-hoc group.
S607
Information Asset permission granted to specific groups and users within Scopeware through either a pop-up list from Scopeware or LDAP
S608
Information Asset permission is frozen not allowing changes by any user.
S609
The Information Asset is available to all public Scopeware users.
S610
The Information Asset is available to select users from public to local groups.
S611
Only specific groups and users within Scopeware can access Information Asset.
S612
Permissions are frozen and do not allow changes to the Information Asset.
Flowchart Shape Key
Process
Document Object
Display

FIG. 6

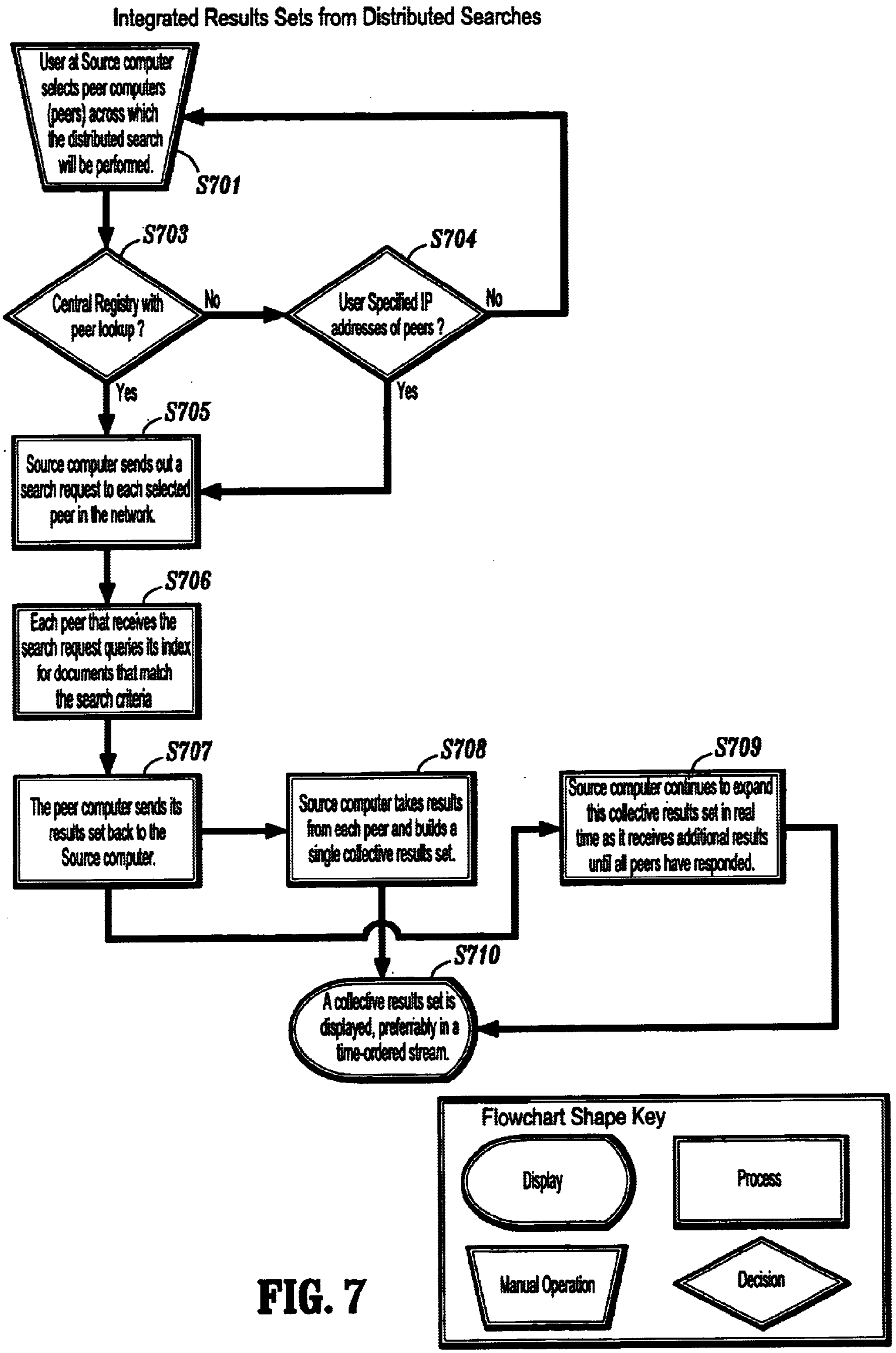

Integrated Results Sets from Distributed Searches
User at Source computer selects peer computers (peers) across which the distributed search will be performed.
S701
S703
Central Registry with peer lookup ?
No
S704
User Specified IP addresses of peers ?
No
Yes
Yes
S705
Source computer sends out a search request to each selected peer in the network.
S706
Each peer that receives the search request queries its index for documents that match the search criteria
S707
The peer computer sends its results set back to the Source computer.
S708
Source computer takes results from each peer and builds a single collective results set.
S709
Source computer continues to expand this collective results set in real time as it receives additional results until all peers have responded.
S710
A collective results set is displayed, preferably in a time-ordered stream.
Flowchart Shape Key
Display
Process
Manual Operation
Decision

FIG. 7

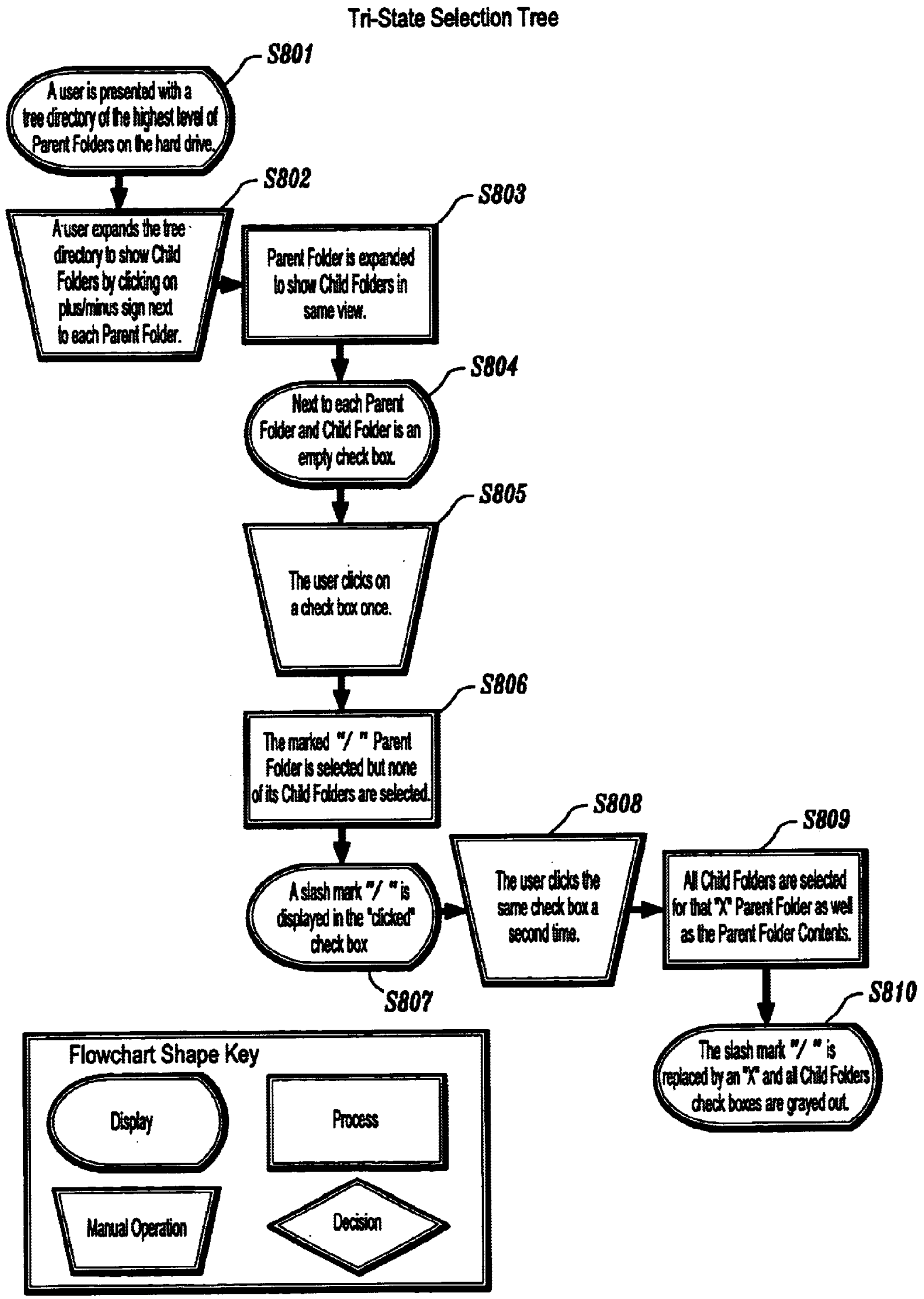

Tri-State Selection Tree
S801
A user is presented with a tree directory of the highest level of Parent Folders on the hard drive.
S802
A user expands the tree directory to show Child Folders by clicking on plus/minus sign next to each Parent Folder.
S803
Parent Folder is expanded to show Child Folders in same view.
S804
Next to each Parent Folder and Child Folder is an empty check box.
S805
The user clicks on a check box once.
S806
The marked "/ " Parent Folder is selected but none of its Child Folders are selected.
S807
A slash mark "/ " is displayed in the "clicked" check box
S808
The user clicks the same check box a second time.
S809
All Child Folders are selected for that "X" Parent Folder as well as the Parent Folder Contents.
S810
The slash mark "/ " is replaced by an "X" and all Child Folders check boxes are grayed out.
Flowchart Shape Key
Display
Process
Manual Operation
Decision

FIG. 8

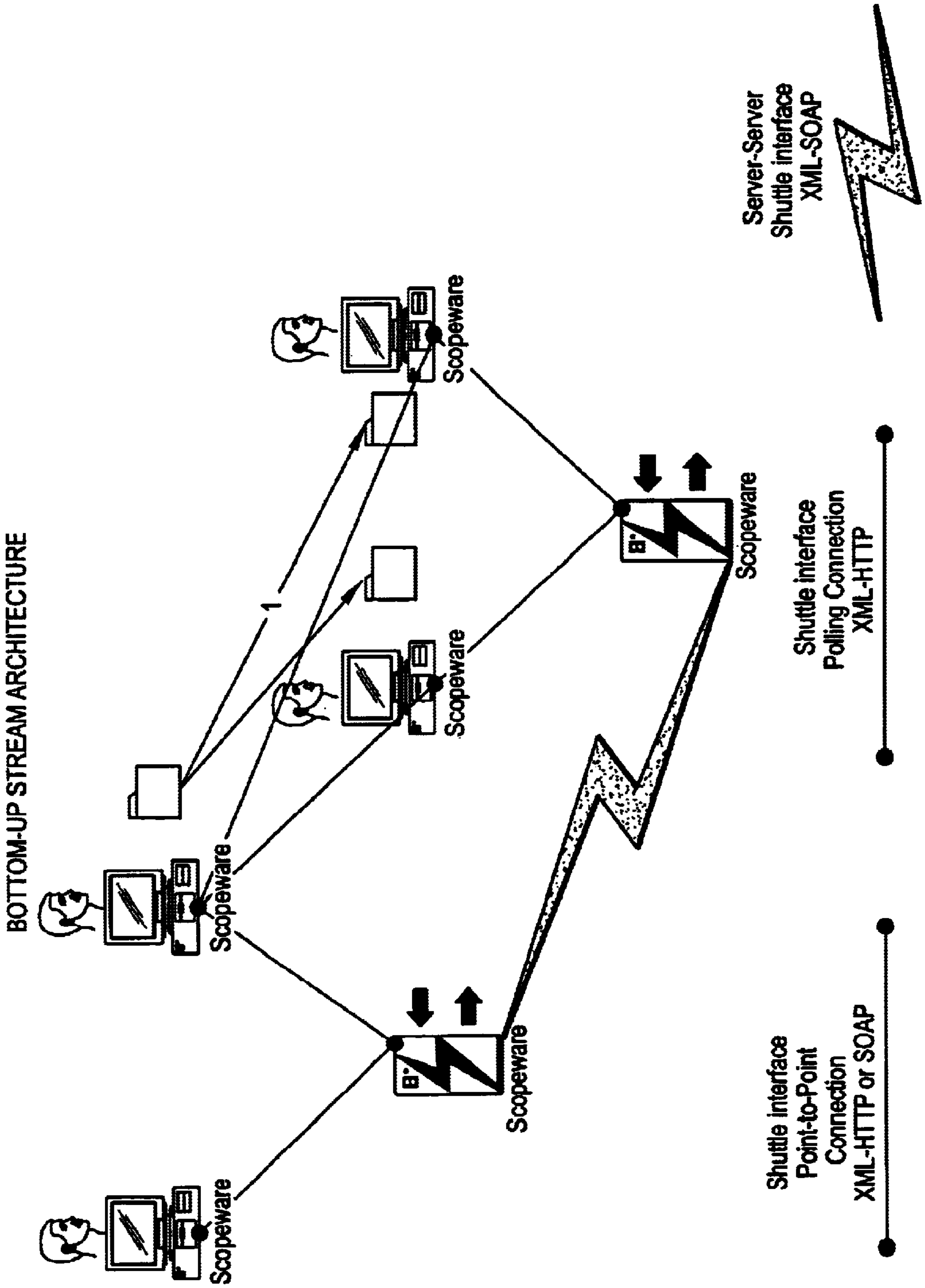

BOTTOM-UP STREAM ARCHITECTURE
Scopeware
Scopeware
Scopeware
Scopeware
Scopeware
Scopeware
1
B'
B'
Server-Server
Shuttle interface
XML-SOAP
Shuttle interface
Polling Connection
XML-HTTP
Shuttle interface
Point-to-Point
Connection
XML-HTTP or SOAP

FIG. 9

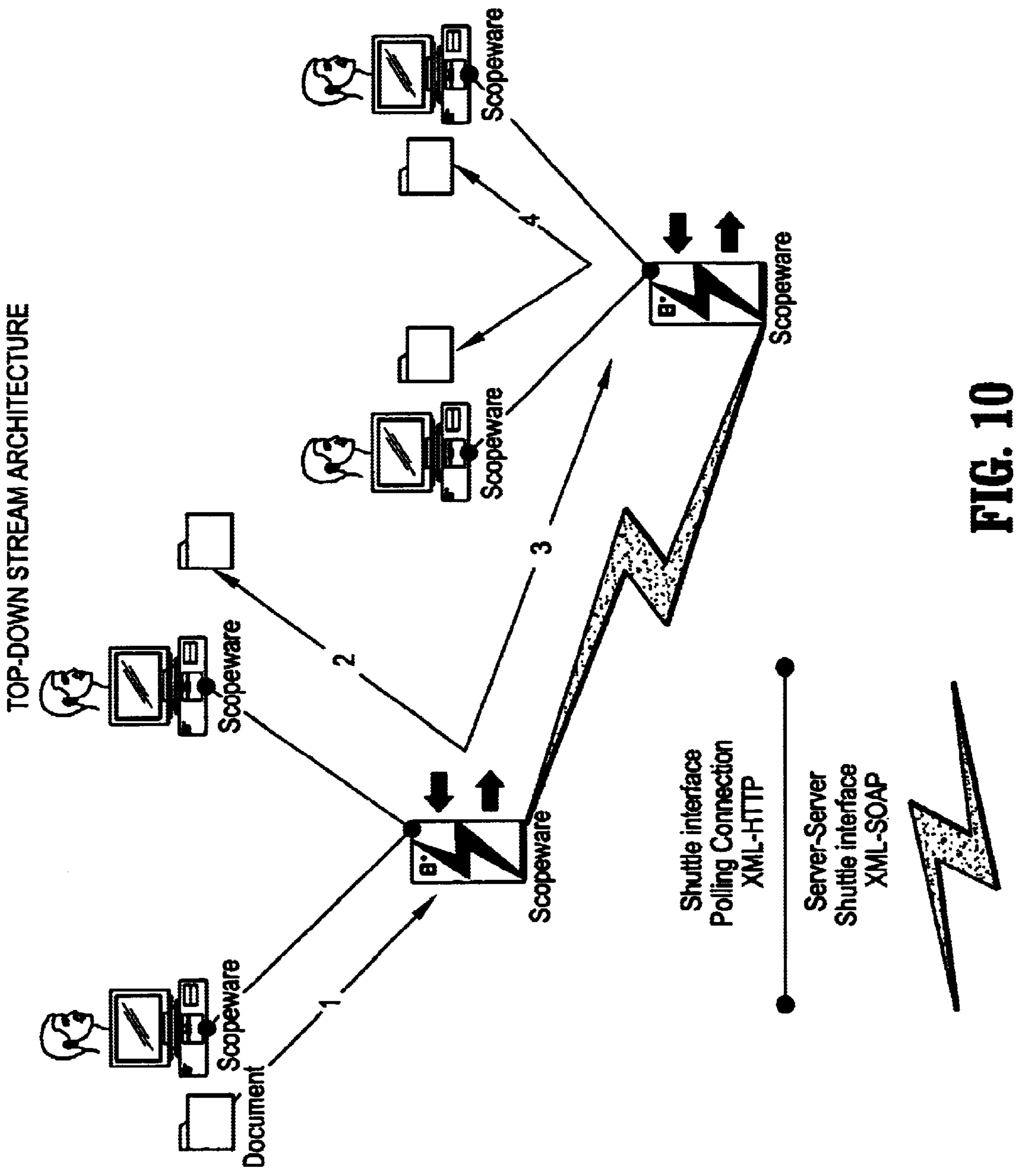

TOP-DOWN STREAM ARCHITECTURE
Document
Scopeware
Scopeware
Scopeware
Scopeware
Scopeware
Scopeware
B'
B'
1
2
3
4
Shuttle interface
Polling Connection
XML-HTTP
Server-Server
Shuttle interface
XML-SOAP

FIG. 10

ENTERPRISE, STREAM-BASED, INFORMATION MANAGEMENT SYSTEM

REFERENCE TO RELATED APPLICATIONS AND INCORPORATION BY REFERENCE

This patent specification (1) claims the benefit of provisional applications No. 60/274,575 filed Mar. 9, 2001 and 60/240,480, filed Oct. 13, 2000 (2) is a continuation-in-part of patent application Ser. No. 09/398,611 filed Sep. 17, 1999, now U.S. Pat. No. 6,638,313 which in turn is a continuation of patent application Ser. No. 08/673,255 filed Jun. 28, 1996 and now U.S. Pat. No. 6,006,227, and (3) hereby incorporates by reference said prior applications in their entireties, as though fully set forth herein.

INCORPORATION BY REFERENCE OF MATERIAL ON COMPACT DISC

this patent specification incorporates by reference the contents of the compact disc attached hereto in duplicate (Copy 1 and Copy 2). Each disc is labeled in accordance with Rule 1.53(e)(6), with the collective names Scopeware 2.0 and vision 1.0. The date of creation of the files on the disc is Jun. 25, 2001. The computer code on the compact disc was generated from correspondingly named source code. The names of individual files on the disc within these collective names, as well as the size of the individual files, are identified in the list of files attached to the Transmission Letter In Accordance with 37 C.F.R. §1.52(e)(ii). The contents of the compact disc submitted herewith in duplicate and the contents of the list of files attached to said Transmission Letter are hereby incorporated by reference in this application as though fully set forth herein.

FIELD

this patent specification is in the field of systems for handling information by computer and more specifically relates to an enhanced system for handling heterogeneous items of information to store, manage, customize, organize and/or deliver such information regardless of its source and type in particularly efficient, easy-to-use, and intuitively understood.

BACKGROUND AND SUMMARY

Traditional information management systems store and retrieve documents on the basis of attributes such as the name and storage location of a document. This, however, can get very unwieldy in typical usage, as more and more names and locations of documents become a part of the storage and retrieval scheme. Although it is possible in some cases to search or order documents by other attributes, such as content and time of creation or revision, it may still be necessary to specify which file folders, directories, or storage devices to search. If a user no longer remembers how a particular item of information was stored in a traditional system, it may be difficult or impractical to retrieve it efficiently.

In an effort to alleviate these and other concerns with traditional storage and retrieval systems, and to provide a more effective and natural approach that better fits the way people tend to work with and think of items of information, a new system described herein uses approaches that rely primarily on an intuitive, time-associated way of dealing with information. The system is stream-based in that it creates time-ordered streams of information items or assets, beginning with the oldest and continuing through current and on to future items. An information item or asset in this system can be any type—a file, an email message, bookmark, IRL, memo, draft, scanned image, calendar note, photo, shopping list, voicemail, rolodex or business card, a video clip, etc. When a user tunes in a stream, ordinarily a receding parade of documents appears on the screen. The closest are nearest in time. When a new document arrives, for example when a new email message comes in, it appears at the head of the stream, at the front of the parade. (When a newer message arrives, it steps in front of the parade.) Further-away documents are older.

Ordinarily, a user stands at the line current in time and looks into the past, but the stream also extends into the future. If the user has a meeting next Tuesday at 10 AM, a note to that effect goes into the stream's future, and a note about a meeting Wednesday goes in the stream in front of the note about next Tuesday. Documents in the stream flow steadily onward, as time does. Documents in the future part of the stream flow toward the present; documents in the present flow toward the past. Newly arriving documents push older documents further into the past.

The receding parade of documents is an efficient way to present information on a computer screen. The display uses foreshortening for a perspective effect to pack more information into limited space. For easy browsing, when the user touches a document on the screen with the cursor, a summary of that document with a thumbnail vies appears immediately, without requiring clicking or other user action, as a browse card—a dedicated small window besides the receding parade of time ordered documents. The user controls the displayed stream with VCR-type controls, to move forward or back, to go toward or to the beginning or the end of time in the stream, to now, or to any date or time, past or future.

An item of information in a stream need not be given a name, or a designation of storage location. In a traditional system, a requirement that all documents have names can have implications beyond the necessity of inventing and remembering names. For example, emails may not have names of their own but may need to be stashed inside some other file; to search for an email the user may need to go to this special mail file and search that file. In the system disclosed here, items of information such as emails do not need to be named and can be searched along with any other types of information items.

Searches in the disclosed system can be by a combination of three methods, search, browse, and time-order.

Time-order in itself often makes it possible to locate documents. Often the user needs a document that showed up recently, this morning, or two days ago, or at some time that can be pinned down with some degree of accuracy. Time-order together with browsing through the stream (and its glance views) makes it possible to glance quickly through the documents that are from the approximate time of interest and quickly pull out the right one. (While traditional systems can time-order documents it often is difficult to intersperse in the list all recent emails, news updates, bulletin-board postings, URLs and other documents, let alone voicemail messages. Without a browse feature for a stream as disclosed herein, such a list can be of little value, whereas with browse and an all-encompassing stream that gets updated promptly with new material, one can sweep over large numbers of documents, get instance glances (summary, thumbnail, etc.) of each and find the right one fast.)

When searching in a stream in the disclosed system, the user gets a new stream—a substream. One can search on any word or phrase, as every word in every document is indexed, on document types and metadata, and on time-related data (e.g., show me all email from last March). If the user searches for an entity called Schwartz Bottling, the new substream will the narrative or documentary history of all dealings with that entity—first contacts, subsequent internal documents or communications, reports, calendar items, and so on.

A substream in the disclosed system is in some ways similar to a folder or directory in a traditional system. Instead of a "Schwartz Bottling" folder in which the user has put documents by so naming them, he/she has created a substream with those document, and can save it for later use or create it again as needed. The substream can do all a folder can but is much more powerful than a folder. A substream collects documents automatically; the use r has to put documents in a folder by hand, one by one. A substream can persist in that it continues to trap newly created or received documents that match it. If a user looks at the "Schwartz Bottling" substream tomorrow, she/he may find it has grown to include a new email or other documents that were interspersed automatically. A substream can tell a story, and include the future. A substream is non-exclusive, in that a document can belong to many substreams. A folder in a traditional system imposes on computers many of the obsolete, irrelevant limitations of a physical filing cabinet drawer or folder. A substream is an organizational tool that can make more efficient use of computer characteristics than an analog of filing an retrieving physical documents.

One reason for the efficiency of the disclosed system is that it handles all types of different documents, or items of information, in essentially the same way, even if the document is of a type or format unknown to the system. Each document when created, received or otherwise encountered is treated consistently according to a universal Document Object Model (DOM). As described below in more detail, the system processes the document to create its Document Object Modes that includes various aids such as significant information about the document including items such as summary, type of document, thumbnail of the document, who is the document' owner, who has permission to access the document, keywords, command options, time stamp, index, etc. This creation of a document's DOM is done automatically, although the user can aid the process. It can be done by a translator agent or programmatically.

The system creates a glance view or browse card of each document that has the same overall format to make searching for and working with a document more intuitive but also is specific to the documents in many ways. One important difference from traditional systems is that the browse card has command buttons that match the type of documents. While the command set for traditional systems may use the same command button set for different types of documents, in the disclosed system the command set that shows in the displayed browse card is specific to the document—it has the unique combination of command buttons that make sense for that document. The command buttons unique to the browse card can be shown on the card itself or separately.

The browse card comes on the screen automatically when the cursor is over the corresponding document in the displayed stream; the user need not take any other action such as clicking on the document or taking an action calling a program that can open or work with the document.

The universal DOM of a document is created automatically as a new document of any type is added to the basic stream of information items. It is done for any existing, legacy documents, when the system is first installed on a computer, and is done as any additional documents are created or otherwise come in. Metadata such as owner, date, access permission and keywords are created as part of this automatic process.

Access permission is a part of a document's metadata, so permission levels need have the constraints of traditional information handling systems where a group or an individual typically has access to all documents in a particular folder or directory, or has a particular type of access to a folder.

Search results are integrated into a substream, at the right place, when and as they become available. The user can start using an incomplete substream and watch it build up. If the search must extend over a number of computers or even servers, and some are unavailable at the time, the results that come in when any become available are integrated into the substream at the right places.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a screen that can serve as a default view when a software product according to a preferred embodiment is opened on a computer; the labels that are added are not normally a part of the displayed screen.

FIGS. **2**–**8** are flowcharts illustrating processes in an example of a preferred embodiment.

FIGS. **9** and **10** are examples of configurations in a preferred embodiment.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

FIG. **1** illustrates a default screen seen on a PC or other equipment working with the disclosed system. It can show up upon turning on the computer, or upon calling the disclosed system. As seen in FIG. **1**, the screen illustrates a receding stream of documents, with the most recent documents at the front. Passing the cursor over a document in the stream causes that document's "glance view" or "browse card" to appear on the screen. The glance view of a document is so labeled in FIG. **1**. The screen also includes the following features appropriately labeled in FIG. **1**: (a) the Search Field is an area in which the user can type one or more words for which the system will search in documents (information assets) in the displayed part of the stream and/or in additional information assets that might not be displayed; (b) the Main Menu is where the user sets preferences, finds help information, logs out, and/or performs other operations; (c) the Header contains information such as links, command buttons and choice boxes used to navigate; (d) the Stream View Options allow the user to configure the presentation of the stream of information assets; (e) the Document Glance allows quick scanning of information assets that are visible on the screen, and presentation of more detailed information on the selected information asset; (f) the Type Glyphs identify the nature of an information asset at a glance (e.g., a Word document); and (g) the Thumbnails is a graphic representation of the type of document (e.g., an audio file, an email, an event, etc.). The User Guide published by the assignee hereof (a copy is submitted concurrently with the filing of this application with an IDS form) further describes the operation of a relevant example and, together with the programs contained in the compact disc submitted herewith, provides a more detailed disclosure of a preferred embodiment.

Certain particularly novel features of the disclosed system are described below by reference to flowcharts and block diagrams. More detailed information on a particular example of implementation of these and other features of the system are evident from the software on the attached compact disc, which is the best mode known to the inventors at the time of filing this patent application.

FIG. **2** illustrates creation of a universal data object model of a documents in accordance with a preferred embodiment. This is an important part of the disclosed system that helps make possible the efficient handling of heterogeneous document types in a manner that users find easy and intuitive. A document object model (DOM) can be thought of as a document shell of the information asset (IA) that contains, anon other items, a thumbnails of the information asset, permission rights, and metadata. The DOM is created from the IA and is stored in a desktop computer and/or a server, either independently of the IA itself or with a replica (copy) of the IA. From there, the system makes the DOM (with a pointer to its IA or replicated IA) to the desktop user or to users that have access to the document through some computer connection.

As seen in FIG. **2**, the process of creating a DOM starts with the uploading at step S**201** of information assets (documents) through a browser or a client software application, or step S**202** with uploading using a software application agent called Doc Feeder in a specific embodiment of the disclosed system. At the following steps, which need not be performed in the order of their description below, a DOM of the IA is created. The IA uploaded at step S**201** or S**202** can comprise structured or unstructured data. At step S**203** the process determines the content type of the IA, e.g., if it is a type that the system recognizes. If it is, the system includes content-type specific metadata in the document's DOM: MIME/content type information, a glyph of the application that creates/views the content-type, and/or the system assigns other content-type data to the DOM shell. If step S**203** determines that the IA is an unknown content type, it assigns to the DOM a content-type for "unknown content-type." Step S**204** extracts text from the information asset, for example, in a text document, this step extracts the text of the document. Step S**205** extracts text that may not be within but may be associated with the information asset, for example, the time stamp of the document, the owner of the document, and possibly other textual information that is or can be associated with the document. Other possible examples are attributes of the IA such as file reference path, database/repository path, file metrics such as size, encryption, other identification information, etc. Step S**206** generates a thumbnail picture of the IA. The thumbnail can be a reduced-size picture of the document, for example of the first page, and can be converted to a graphic image format. Other examples of thumbnails are JPEG, MPEG, BMP, GIF, AVI, or other still or moving image files representative of some aspect of the IA. Step S**207** produces an automatic summary of the IA, e.g., a replica of its first 500 words, or first 10 sentences, or some other information copied or otherwise derived from the IA. Step S**208** creates a permission list unique to the IA that defines the owner of the IA (e.g., its creator), and lists of people or entities and groups that can access the IA or the DOM of that IA for reading and/or writing purposes. This permission list can be defined by the user for the particular IA or for a class of IAs, or can be created automatically, e.g., by software agents called Doc Feeder or Crawling agent in a particular embodiment of the described system, or by programmatic mapping such as LDAP, Active Directory, NTDS or some other mapping. Alternatively, at least for some documents, the permission list can be default setting.

Step S**209** assigns keywords to the information asset. The software agents Doc Feeder or Crawler can assign keywords, and the user can manually assign or add keywords. Step S**210** generates and assigns to the IA a Globally Unique Document ID, e.g. as 64 bit code unique to the IA. Step S**211** determines and assigns to the IA document operations that are unique to the IA. Depending on the IA, these operations or command buttons can be basic, such as "View" and "Reply." They can be content-specific, such as "Play" for multimedia information assets. They can be solution-specific, such as "Fax" of Purchase." They can be user-specific, such as "Delete" allowed to only certain users. An important point is that the operations or command buttons assigned to a particular IA match the IA and need not be the same for different information assets, as is the typical case with traditional information management systems. Step S**212** assigns optional operations or command buttons to the IA. They include, for example, commands to send the IA to an optical character recognition (OCR) service that can be a separate service, IP, HTTP-based or an asynchronous operation. Alternatively, the optional operation can be another OCR operation that can perform OCR on a selected part of the IA, or on digital graphic portions or can involve multipart associations. At step S**213**, the information asset is submitted to an indexing engine (asynchronous service) Again, this can be a separate service, IP, HTTP-based. This step can index all or selected fields of the IA, including but not limited to the IA summary, title, permissions, IA text, keywords, time, metadata, and content-type. At step S**214** the DOM created as described above is submitted to a storage service. This can be a database that is a file reference with a pointer to the actual location of the IA on a network or a local file system, or it can be a database that contains the actual IA in a repository such as a user's computer or a centralized repository. The document object model so generated is made available for use in step S**215**.

FIGS. **3** and **4** illustrate methods of creating document object models from information assets. As seen in FIG. **3**, three type of information assets are involved—new information assets **301**, modified information assets **301**, and deleted information assets **303**. All come to a file system **304**. At step S**305**, agents specific to the disclosed embodiment of the system known as Scopeware 2.0 translate the IA into a DOM, i.e., create a DOM shell for the IA, with attributes as discussed in connection with FIG. **2**. At step S**306**, Scopeware agents translate the IA modifications into an updated DOM and time-stamp the change so the new time-stamp becomes a part of the DOM and the modified IA can be places in the stream of documents at a place reflecting the new time-stamp. At step S**307**, Scopeware agents execute actions for removing the deleted IA from the repository of documents. The display, such as that seen in FIG. **1** reflects the actions takes at steps S**305**, S**306** and S**307**. As a result of step S**305**, the stream on the display shows at 308 the new IA (provided the time period where the new IA fits is being displayed). As a result of step S**306**, the modified IA appears at **309** in its correct place in the displayed receding stream of documents. As a result of step S**307**, the deleted documents is removed at 310 from the displayed stream, and the remaining. In FIG. **4**, a programmatic information system received new, modified and deleted information assets for storage and distribution to appropriate translation agents as illustrated. In other respects, the FIG. **4** arrangement corresponds to that of FIG. **3**, so the description of corresponding portions will not be repeated.

At least some of the document object model created as described above becomes a part of a glance view or browse card of the type illustrated in FIG. **1**. An important feature of the system disclosed here is to conveniently display such a glance view in a natural and intuitively accepted way to facilitate operations.

Traditional user interfaces for computers typically present lists or graphical icons of "documents" (including but not limited to computer files, emails, web pages, images and other types of electronic information). These lists and icon displays provide only a limited amount of information about the document—typically, title and application type only, although additional information as well in some cases. This can make it difficult for users to identify the document without downloading and/or opening the document with its associated application. For example, in Windows 2000, the user interface displays a small temporary pop-up window of the document's title, application type, author and size when the user hovers his cursor on the document icon; however, the pop-up window appears only after a brief delay, usually 1–2 seconds and is for documents that are on the screen at the time, which tend to be a small part of the many documents typically stored in or accessible through a user's computer.

In contrast, the disclosed system creates a pop-up window for heterogeneous documents of known and unknown application types that appears instantly, as perceived by the user, as he/she hovers the cursor over the document's representation in the user interface. In the example of FIG. **1**, this representation is an index card in a cascading flow of overlapping index cards (called "browse cards"), and the pop-up window is called a "glance view". This glance view not only contains the document's title, application type and owner, but also may contain rich multimedia cues (such as a thumbnail image of the first page of the document, a WAV or MP3 preview of an audio file, or an animated GIF preview of a video file), text summaries and document operations specific to the document's application type and access permissions. For example, if the user has write permission for a document, the "Edit" operation will be visible and available; however, if not, the Edit operation will not be visible or available. These document operations are interactive, allowing users to select available operations directly.

Referring to FIG. **5** for an illustration of the instantaneously dynamic, tailored, and interactive document glance view feature of the disclosed system, at S**501** a user hovers his or her computer cursor over a document's browse card. Essentially instantly, at least as perceived by the user, and without any mouse clicking or other action on the part of the user, step S**502** processes the information needed for a glance view to appear on the screen, and at S**503** the glance view appears next to the browse card, using a technology such as Dynamic HTML. If the user clicks on a document's browse card, as detected by the test at step S**504**, and as executed by the user at S**505**, step S**506** causes the glance view to become fixed and step S**507** causes it to remain in the display. The glance view does not change until the user clicks on another document's browse card. If the user does not click on any browse card, as determined by the test of step S**504**, the glance view will instantly change as the user moves his cursor over other browse cards, to reflect the glance view of the underlying browse card. If the user has clicked on a browse card to fix the glance view as a stationary window, the user can then select any of the visible and available document operations, by taking the "yes" branch of step S**508** and selecting at S**509** an available operation (as earlier described, the operations or command buttons that show are specific to the document). At step S**510** the system executes the selected operation (command) and the display reflects this at S**511**. If at step S**508** the user takes the "no" branch, she can continue ro hover the cursor over the stream of browse cards and repeat the process, at step S**512**. If at S**504** the system determines that the user has not clicked to fix a glance view, the glance view information essentially instantly changes at S**513** as the user moves the cursor over other browse cards, and the new glance views appear on the screen at S**514**.

FIG. **6** illustrates a process involving another important feature of the disclosed system—granular permissions for access to information assets that allows clients to receive seamless and uniform access to contents without necessitating changes to existing network security and access rights. In traditional systems, a network administrators typically would grant access to specific network drives and file folders. The permission typically would allow a user to access the entire folder or drive, or would deny access to an entire folder or drive, rather than to a particular information asset or document.

In the disclosed system, each information asset is accessible through specific access permission for each client or designated group of clients. Examples of access stage permissions are read, write, and aware. Read permissions allow a client to view the full information asset. Write permissions allow the client to view and edit the document. Aware permission alerts the client that an information asset exists, for example by providing a document shell in the client's stream of documents, but does not allow the client to view or edit the document. A group of clients who want to collaborate on a project or event can establish a designated group that can be assigned permissions to relvant documents for the project or event. Thus, each member can receive real-time additions to his or her stream of documents and information assets are posted. The clients can assign permission to the other group members themselves, by so designating the appropriate documents to be shared, without involving a network administrator. Some documents, such as personal to-do lists, can be accessible only to a specified user, but the user can change this at any time to allow access, full or partial, to other designated persons. Assignments of permissions for access can be done as granularly as an individual client level or individual document, or as diffuse as a departmental or enterprise level.

As seen in FIG. **6**, an information asset **601** can have permission levels assigned to it in several ways. At step S**602**, a software agent such as Doc Feeder can automatically assign permissions; at step S**603** a programmatic system such as SDAP, Active Directory, Access Control Lists, NT DS, of some other system assigns permissions to the document; and/or at step S**604** the user manually assigns permissions to the document. Examples of processes relevant to different types of permissions are: step S**605** grants access to all public users of the system; step S**606** assigns permissions to groups as illustrated; step S**607** assigns permissions to specific groups as illustrated, and step S**608** freezes permissions and does not allow the document to be changed. The display, of the type illustrated in FIG. **1**, can provide information representative of the permissions, as illustrated at steps S**609** thorugh S**612** in FIG. **6**.

Another important feature of the disclosed system is illustrated in FIG. **7** and pertains to integrating search results from distributed searches. In traditional systems, search requests in a client/server model with a central index usually return a single, well-defined results set. In a peer-to-peer network, however, search results may come back to the "Source" computer (the computer that issues the search query) in a haphazard manner because of network latency (variable traffic speed and bandwidth across a distributed network) and variable peer presence (peer computers can be turned on and off, or removed from network at times).

The disclosed system asynchronous responses to a distributed query across a peer-to-peer network of computers to integrate the results from diverse sources, arriving at different times, and comprising diverse types of documents, into a single unified results set. One preferred embodiment leverages the time-ordered presentation interface earlier described in so that search results are integrated into a time-ordered stream according to each document's original time-stamp, regardless of when the document's search results set was received by the Source computer.

As seen in FIG. **7**, at step **701** a user at a Source computer selects peer computers ("Peers") across which the distributed search will be performed. If the test at S**703** determines that there is no central registry with peer hookup, and the test at S**704** determines there is no user-specified IP address of peers, the process returns to S**701**, where the user can specify addresses or they can be provided in some other way. The central registry with lookup of Peers can involve Online/offline status, IP/DNS resolution service and Optional public/private key authentication. When the test at S**703** or at S**704** leads to the "yes" branch, at step S**705** the Source computer sends out a search request that travels to each selected Peer in the network. At S**706**, each Peer that receives the search request queries its index for documents that match the search criteria, and at S**707** the peer computer then sends its results set back to the Source computer. The response can be XML-based, a binary byte stream, or an in-band and out-of-band transfer. At S**708** the Source computer takes the results set from each Peer and builds a single collective results set. In a preferred embodiment, this collective results set is organized as a time-ordered stream of documents, as seen in FIG. **1**. This can involves an on-the-fly browser combination with XML & XSL with time-sort algorithm, XML to presentation layer with time-sort algorithm, and in-band and out-of-band transfer. Improtantly, at S**709**, the Source computer continues to expand this collective results set, essentially in real time as it receives additional results sets from Peers until all Peers have responded or some other relevant event has taken place. At S**710**, the collective results are displayed as soon as results have come in at the Source computer, and the display is updated as additional results come in, even when a Peer that was off-line comes on line and sends results at a later time.

Yet another feature of the disclosed system is a particularly convenient tri-state tree. In a single scrolling tree directory of the contents of a hard drive (or hard drives in a network), a user may want to select "Parent Folders" (folders containing subfolders) and "Child Folders" (subfolders contained within a folder) that can be further operated on. This feature allows users to select folders in one or more of the following combinations:

1. All Parent Folders and all Child Folders
2. Some Parent Folders and all their Child Folders
3. Some Parent Folders and some of their Child Folders
4. No Parent Folders and no Child Folders (the do nothing option)

This selection tree has useful application beyond the particular example of information handling disclosed here; it can be used to select folders for any computer operation. For example, it can enable users to discretely select software application or operating system components to install or remove.

A single scrolling tree directory of Parent and Child Folders that can expand and contract to show the contents of Parent and Child Folders is known—Microsoft Windows Explorer is an example of one. A Tri-State Selection mechanism also is known—Microsoft Add/Remove Windows Components is an example of another way of selecting various Parent and Child Folders. However, the Microsoft Add/Remove Windows Components feature does not display all Parent and Child Folders within a single scrolling tree directory; Child Folder and other contents of a Parent Folder are displayed in a separate window only after the user clicks on a Details button. In addition, only the contents of one Parent Folder can be displayed at a time.

The Tri-State Selection Tree described here combines the elements of a single scrolling tree directory with a tri-state selection mechanism in a new and unique way to enable users to discretely select specific Parent and/or Child Folders all in one single view.

Referring to FIG. **8** for an illustration, at step S**801** a user is first presented with a tree directory of the highest level of Parent Folders on a hard drive or network. At S**802** the user can expand the tree directory to show Child Folders by clicking on a plus/minus sign next to each Parent Folder, and the directory so expands at S**803**. At S**804**, the display shows a check box next to each Parent Folder (e.g., to the right of the plus/minus sign). By default, all check boxes are empty, indicating that no Parent or Child Folders are selected. If at step S**805** the user clicks on a check box once, the process at step S**806** selects the marked "/" Parent Folder but none of its Child Folders are selected, and step S**807** shows this on the display. If at step S**808** the user clicks the check box a second time, the slash mark is replaced by an "X" and all the Child Folders' check boxes are then selected and grayed out at S**809**, indicating that all Child Folders are selected for that Parent Folder, and this is displayed at S**810**.

Thus, by expanding the tree and clicking on check boxes, the user can systematically and efficiently select a discrete number of folders on which to perform an operation.

Yet another feature of the disclosed system is an arrangement of a redundant array of inexpensive servers (RAIS). Processing of a large set of information or document requires benefits of a centralized architecture—reliability and scalability, and RAIS is a novel approach to provide benefits of a centralized architecture—namely reliability and scalability with numerous inexpensive computers. Thus, RAIS can deliver essentially infinite scalability, can allow inexpensive smaller computers to be used to solve enterprise computational problems rather then expensive larger platforms, cheaper/faster.

For example, consider:

Set of Information, D, with specific documents D**1**, D**2**, D**3**; D{D**1**, D**2**, D**3**}

RAIS of N×N size here with N=3; RowN,CoIN

Replication factor is number of columns

Scalability factor is number of rows

1. Here N=3, with 9 computers

	Col1	Col2	Col3
Row1	A	A	A
Row2	B	B	B
Row3	C	C	C

2. To post a Document, Dn, one copy is sent to a sub-server in each CoIN, so

	Col1	Col2	Col3
Row1	A(Dn)	A(Dn)	A(Dn)
Row2	B	B	B
Row3	C	C	C

3. Thus Dn is replicated N times (N=3) and thus if Col**1**:Row**1** computer is unavailable there are two other computers with the same Dn. This is RAIS replication.
4. To post a universe, or set of documents, D{D**1**, D**2**, D**3**}, can use simple (round-robin) or complex (latency, closest path, spanning tree) routing, sending each document to a different RowN.

	Col1	Col2	Col3
Row1	A(D1)	A(D1)	A(D1)
Row2	B(D2)	B(D2)	B(D2)
Row3	C(D3)	C(D3)	C(D3)

5. Thus to reassemble the entire universe or set of documents, D, need to send a request to each RowN. To reconstruct, D, for an N×N RAIS requires N request/responses.
6. Multiple smaller requests can be used instead of one mammoth request. This reduces latency, bandwidth and process constraints. This is RAIS scalability.
7. Note that any one of the computers in Row**1** can be used to re-construct the total set D found in Col**1**. For example, if Row**1**:Col**1** computer is unavailable, then Row**1**:Col**3** computer has a copy of the data. In fact, D is can be constructed from any arrangement that completes a CoIN.
8. To increase either replication or scalability simply increase N.

Scopeware Software Agents, either desktops or servers, can be installed on each computer in a RAIS matrix to achieve this functionality.

The disclosed system can be implemented in a variety of ways in terms of physical information storage—for example, physical information storage can be centralized or decentralized. Decentralized storage, physical storage of information with multiple servers and/or clients, is possible through network agents called Doc Feeders, which may be located at a server or client level. The Doc Feeder allows a storage location of a client, for example a file folder on a desktop hard drive, to be included in the system level data repository for use throughout an organization or enterprise. Depending upon implementation, the Doc Feeders can replicate the information asset (IA) to a server or maintain a constant pointer to the physical storage location while populating the system with the document object model (DOM). As earlier described, a DOM is a document shell of the IA that contains, among other items, a thumbnail of the IA, permission rights, and metadata. A DOM is created from the IA and placed on the Scopeware server, either independent of the IA or with a replication of the IA. From there, the Scopeware server will share the DOM (with constant pointer to the IA or replicated IA) with other connected system servers and clients in order to make the IA available to all clients connected to the network. Thus, the system servers and network agents (Doc Feeders) act as document proxies for both storage and retrieval of IAs.

In addition, the system servers within the network need not be physically close in proximity. For example, a client in a truly global organization with locations and system servers on several continents can query and retrieve sales results across all system servers and clients through a federated search. In essence, the disclosed system creates a virtual store from all documents accessible to any system server or client either centralized or decentralized.

The physical information storage of the disclosed system follows three models: duplication, replication, and document reference. The duplication model physically stores a duplicate IA on the parent Scopeware server that was created by the client. Other clients polling the parent Scopeware server have full access to the IA, depending upon permissions, whether or not the original document is available from its native storage location (i.e. client PC is turned off). The replication model replicates the IA from the parent Scopeware server to the peer Scopeware servers within a federated network. All clients within the federated network have full access to the IA, depending upon permissions, whether or not the original document is available from its native storage location (i.e. client PC is turned off). An example of the replication model is the concept of a redundant array of inexpensive servers. This concept, which is described in detail in the distributed enterprise model, utilizes client machines in place of a singe server. The document reference model "parks" only a DOM of the IA on all Scopeware servers and maintains a constant pointer to the actual physical location of the IA rather than storing a full copy of the IA on the Scopeware server. Other clients will only be able to gain access to the IA when the physical location of the IA is connected to the network (i.e. client PC is turned on).

There are to primary types of streams in accordance with the disclosed system: Bottom-Up and Top-Down. Through the use of both Bottom-Up and Top-Down methodologies, Scopeware creates a living stream for the client with new DOMs appearing automatically as content arrives. The Scopeware distributed enterprise model can make use of both server-based resources and client-based resources where appropriate. Both types of streams can be used simultaneously and interchangeably.

Bottom-Up streams are comprised of information collaboration formed by ad-hoc groups of Scopeware clients. A bottom-up stream is composed of information created by the clients of a transitory group. Information shared and created by this group is be replicated via point-to-point connections (i.e. from client PC to client PC). In this way, bottom-up groups can form and disperse frequently, and without notification, while its members will still have access to the shared information. FIG. **9** illustrates this configuration.

Top-Down streams are more permanent, generally more administrative streams or collections of information, such as company-wide distribution lists, or groups like 'Accounting' and 'Development'. In these groups, information is "parked" to the server from the desktop. The server then sends the information to other known servers. Each client maintains a polling connection to the server to retrieve "parked" documents that have recently arrived from other remote servers or from local clients. FIG. **10** illustrates this configuration.

As earlier described, the user interface within the Scopeware product portfolio has unique characteristics. The DOM provides certain information that allows quick perusal of the information retrieval results via a proprietary "browse card" or "glance view" which is similar to an index card that contains data on the underlying IA. A unique "browse card" or "glance view" is created for each IA. The "browse card" or "glance view" includes metadata for the document, which is comprised of a title, identification number unique to Scopeware document referencing, date/time stamp, and owner information. The "browse card" or "glance view" also presents a thumbnail image of the IA and a summary of the IA contents. Finally, the "browse card" or "glance view" contains a list of operations appropriate for the IA's application that include, but are not limited to, copy, forward, reply, view, and properties.

The "browse card" or "glance view" arrives in the stream of those clients that have permission to view the IA. The owner can grant access to other clients or groups by granting read, write, or aware permissions through the properties of the "browse card" or "glance view." Permission can be granted as granular as an individual-by-individual basis from the DOM, or through predetermined administrative groups via the Scopeware server.

The "browse card" or "glance view" is presented in a time-ordered sequence starting in the present going back into the past. The "browse card" or "glance view" is available in a number of views. The primary view is the stream. Other formats include a grid, Q, list, and thumbnails. The various views address the client's personal preferences for accessing time-ordered content in their most logical way. These views all contain the information presented in a "browse card" or "glance view" but are organized in a different method. Other specialized views include the address book and calendar.

An advantage of the "browse card" or "glance view" approach is the ease of browsing, searching, and retrieving IAs. In the stream view, the "browse card" or "glance view" of each IA are aligned much like cards in a recipe box. For each item, the title and application icon are viewable on the "browse card" or "glance view" in the stream. When the client passes over the "browse card" or "glance view" in the stream with the mouse pointer, the full "browse card" or "glance view" is presented to the client for easy viewing. From the "browse card" or "glance view," the client can perform any of the aforementioned actions available to the IA, subject to permission access.

The disclosed system is suitable for a number of computing models servicing multiple clients including a single departmental server model, an enterprise server model, a distributed enterprise model, and a peer-to-peer model (absent a dedicated Scopeware server or common server). In addition, the software enables wireless computing independent of or in conjunction with any or all of the aforementioned models. Wireless clients include WAP enabled phones, PDAs, Pocket PCs, and other similarly capable devices capable of receiving and transmitting data across a network. All of the Scopeware Implementation Models make use of the components previously discussed, providing consistent interface available across different computing topologies, from monolithic single servers to peer-to-peer collaboration.

Access to the IA contained in the Scopeware repository can be achieved through two methods. The first method of access is through the thin-client method. The thin-client method utilizes a web browser, such as Microsoft's Internet Explorer or Netscape's Navigator, on the client device to gain access to the Scopeware repository residing on the Scopeware server. The second method of access is the desktop-client method. The desktop-client method involves a local installation of Scopeware on the client device. The client device is then capable of performing the storage, retrieval, extraction, and processing of IAs as they are introduced to the Scopeware repository. All the models below can utilize either method of access to the Scopeware repository, however the distributed enterprise and peer-to-peer models are optimized with the desktop-client method.

Single Server Model. A single server model makes content on one Scopeware server available to any client connected to the departmental server. The Scopeware software creates a unique DOM that represents to the user interface the relevant details of the IA physically stored by the server or client. Thus, when a client connected to the network requests access to and retrieval of IAs through Scopeware, the client can view all documents contained within the network that satisfy the query parameters and access restrictions regardless of the document's native application. The documents available include those stored locally by the client, those saved to a central storage location, and those stored by peer clients with Doc Feeders connected to the shared server.

Enterprise Server Model. In an enterprise server model, where multiple Scopeware servers are installed, federated access to and retrieval of IAs across the network is enabled. In federated information sharing, a client asks one Scopeware server for IAs that may reside on it or one of many connected peer Scopeware servers. In this model, the actual IA may reside on any network-connected client, the Scopeware server, or a centralized data storage location. Transparent to the client, the Scopeware servers shuffle the retrieval request and access restrictions to present a single, coherent stream to the client via the presentation architecture previously discussed (within the original patent document).

Distributed Enterprise Model. A distributed enterprise model utilizes the clients for storage, retrieval, and processing of IAs. Through the use of directory monitoring agents, similar to network agents, the physical location of an IA need not be on the Scopeware server, but rather can reside with any client. The Scopeware servers take on a secondary role as administration servers and content parking lots. This model pushes the processing tasks to the clients while using the servers to shuttle IAs throughout the enterprise. The indexing engine, thumbnailing engine, lightweight storage database will be based at the clients.

Taking Scopeware beyond distributed networking and the federated architecture—into a more distributed approach will be straightforward, given the way that the system has been designed. Key elements of the next stage of deployment are distributed document processing and scalable server arrays.

Distributed document processing consists of two different approaches. First, when information was created physically on a desktop machine, but was part of a larger application and intended for storage on a server (rather than on the desktop), the Desktop facilities could do the document extraction, indexing, thumbnailing, etc., and post the results to the Scopeware Server. Second, a Scopeware Server that was handed a document (perhaps from an OCR process or from a central email application) could hand the document off to an available Scopeware Desktop for the same processing. These strategies relieve the processing load on the Scopeware Server and leave it free to focus on handling searches and stream integration, allowing a given Scopeware Server to handle a much larger user load.

When an organization needs to support central processing of large document bases—and needs the reliability, accessibility and security of a centralized architecture—Scopeware Servers will support deployment in a novel architecture we have named RAIS—a redundant array of inexpensive servers.

In this architecture, imagine a square array of desktop machines—call each one a "sub-server." The array as a whole comprises the Scopeware Server. (This does not require wiring together an actual array or cluster; any interconnect such as a Ethernet sub-net or even HTTP over a broader network will work.) In these arrays, columns of servers provide redundancy for storage, while rows (within columns) provide redundant points of distribution.

To post document D, one copy of D is sent to a sub-server in each column of the array. To replicate everything five times such that losing any data requires the loss of five sub-servers, five columns are used. The number of columns in the array is managed to support exactly the degree of replication (and redundancy) desired. The write processes can be managed in a number of ways to ensure that the different rows in the columns are balanced.

To send a polling message or search request ("give me all the latest stuff"), a request is sent to each sub-server in one column (note that the means to do this transparently to the user is an extension of the federated search technology). Each column of sub-servers absorbs one copy of every posting (because any write has gone into at least one row of the column); therefore, all the sub-servers in any one column collectively have copies of everything. Just a "replication factor," is chosen for data redundancy, a "distribution factor" is chosen for responsiveness and for data management, representing the number of rows in any column. To get ten small responses to a search request instead of one big response, or to distribute the total data-storage burden over ten machines instead of one, the array is implemented with ten sub-servers in every column.

The entire "Server" can be run with only one row (resulting in replication, but no distribution) or with only one column (resulting in distribution but no replication). In the limit, row size=column size=1, and the effect is to have a single conventional server.

This approach to distributed processing, scalability and reliability for large applications allows arbitrary sets of "smaller" computers (single/dual processor, inexpensive memory and disk storage) to be used in place of very large, expensive machines. This allows the application platform to be designed to the reliability and access requirements of the particular application, and then scaled incrementally (by adding more small machines into the array) as the actual application grows in terms of users served or information managed.

Distributed document processing and server arrays will give Scopeware almost infinite scalability while maintaining compatibility with early solutions or architectures. In addition to adding greater reliability, this architecture will support very large information processing applications. This will allow enterprise-scale, top-down applications—inbound support/sales email handling, customer service or even IRS-scale tax document processing.

Distributed document processing (with Scopeware Desktop) could be combined with either a "conventional" (1 processor array) Scopeware Server or with a more powerful array. This will allow organizations to create departmental or workgroup level solutions that can grow into enterprise applications if necessary.

At the same time, the system will allow users themselves to create self-organizing applications based on their specific and current needs. Ad hoc teams can create collaborative spaces that cross organizational boundaries if necessary. These applications can leverage either Scopeware Desktops or departmental-level Scopeware Servers.

Because the system has the architecture and capacity to support any level of centralization or decentralization concurrently, applications and their platforms can be engineered centrally or grown organically, and they can be tailored to the needs of their users and the organization on an ongoing basis.

Peer-to-Peer Model. The peer-to-peer (P2P) model allows multiple clients to share IA directly without the use of a dedicated Scopeware server. The P2P model allows for pure ad hoc collaboration among Scopeware clients. For example, a client can share IA via the Internet with identified Scopeware clients that have permission to access IA from the client, and vice versa. This is similar to the distributed enterprise environment except the dedicated Scopeware server has been removed as a storage, retrieval, and connection mechanism. Instead, Scopeware clients will connect point-to-point with other Scopeware clients through a general network connection such as the Internet.

Using P2P, a client can create a virtual shared stream that looks as though it is stored on a server but is in fact stored only by many clients. Historically, all clients would need access to a shared file folder on a common server in order to share information. With Scopeware, clients can share information that is located on each other's device and are not restricted to a common server or single physical storage location. To illustrate, five clients of Scopeware want to create a shared virtual stream to support a project. They call their group "Team One." Then, when any member of "Team One" posts a document to his or her stream, and marks it "readable by Team One," the system automatically sends a copy to every Scopeware client on the "Team One" list. Each Scopeware client receiving this document pops it into its client's local stream. Thus information created by a client who is a member of "Team One" (and flagged for Team One by the owner) winds up in the local stream of every member of Team One, whether the post is a document, an event (team meeting), task, or contact. It's as if he had sent his posting to a "client" server, and then everyone had polled the server, but in fact there's no server.

What is claimed is:

1. A method of operating an enterprise information management system comprising at least one server and a number of personal computers selectively communicating with each other comprising:

creating document object models comprising selected information from and about information assets of diverse types, created by diverse software, said document object models having a consistent structure;

displaying browse cards related to respective ones of the information assets in a time-ordered stream, together with glance views related to the document object models of the respective displayed documents, said glance views being displayed essentially in real time in response to passing a cursor over respective ones of the browse cards.

* * * * *